JBoss WS User Guide

Version: 1.1.0.M2

JBossWS Runtime Overview

JBossWS is a web service framework developed as a part of the JBoss Application Server. It implements the JAX-WS specification that defines a programming model and run-time architecture for implementing web services in Java, targeted at the Java Platform, Enterprise Edition 5 (Java EE 5).

JBossWS integrates with most current JBoss Application Server releases as well as earlier ones, that did implement the J2EE 1.4 specifications. Even though JAX-RPC, the web service specification for J2EE 1.4, is still supported JBossWS does put a clear focus on JAX-WS.

1.1. Key Features of JBossWS

For a start, we propose you to look through the table of main features of JBossWS Runtime:

Table 1.1. Key Functionality for JBossWS

Feature	Benefit
JAX-RPC and JAX-WS support	JBossWS implements both the JAX-WS and JAX-RPC specifications.
EJB 2.1, EJB3 and JSE endpoints	JBossWS supports EJB 2.1, EJB3 and JSE as Web Service Endpoints.
WS-Security 1.0 for XML Encryption/Signature of the SOAP message	WS-Security standardizes authorization, encryption, and digital signature processing of web services.
JBoss AS	JBoss Application Server 5 (JavaEE 5 compliant) web service stack.
Support for MTOM/XOP and SwA-Ref	Message Transmission Optimization Mechanism (MTOM) and XML-binary Optimized Packaging (XOP) more efficiently serialize XML Infosets that have certain types of content.

Creating a Web Service using JBossWS runtime

In this chapter we provide you with the necessary steps to create a Web Service using JBossWS runtime. First you need to create a Dynamic Web project:

2.1. Creating a Dynamic Web project

Before creating a web service, you should have a Dynamic Web Project created:

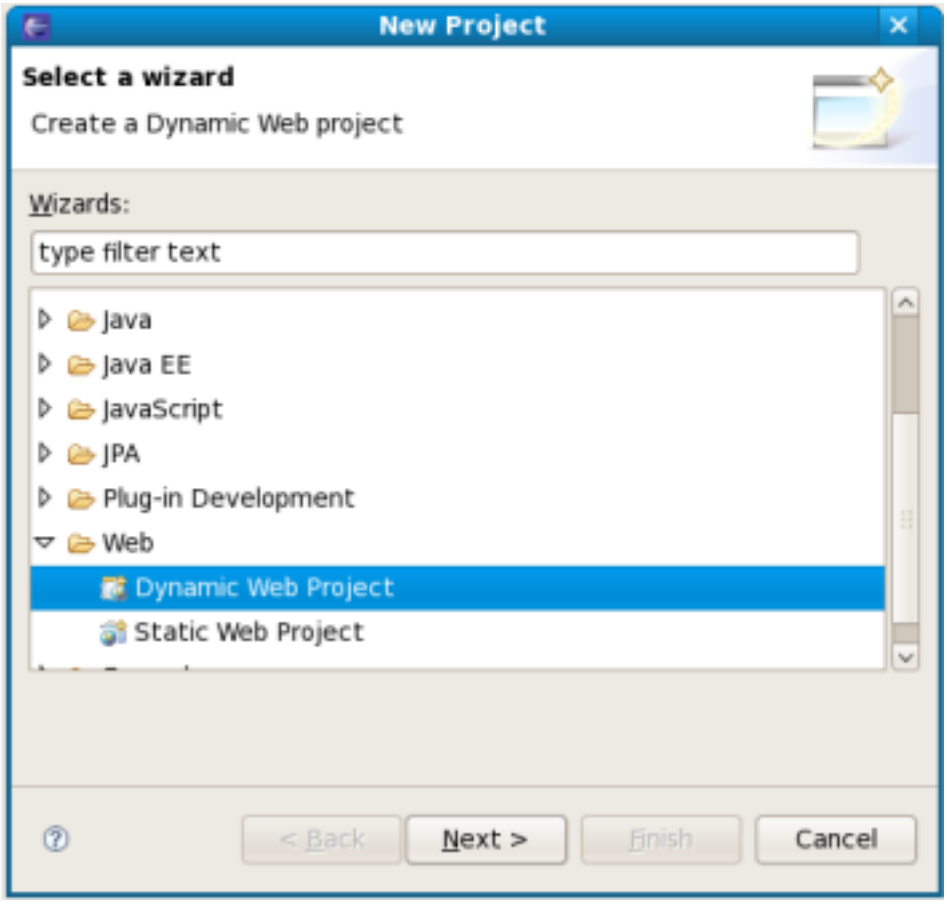

Figure 2.1. Dynamic Web Project

Create a Web project by selecting *New > Project... > Dynamic Web project*. Enter the following information:

- Project Name: enter a project name
- Target runtime: any server depending on your installation. If it is not listed, click New button and browse to the location where it is installed to. You may set *Target Runtime* to *None*, in this case, you should add *JBoss Web Service facet to the project*.

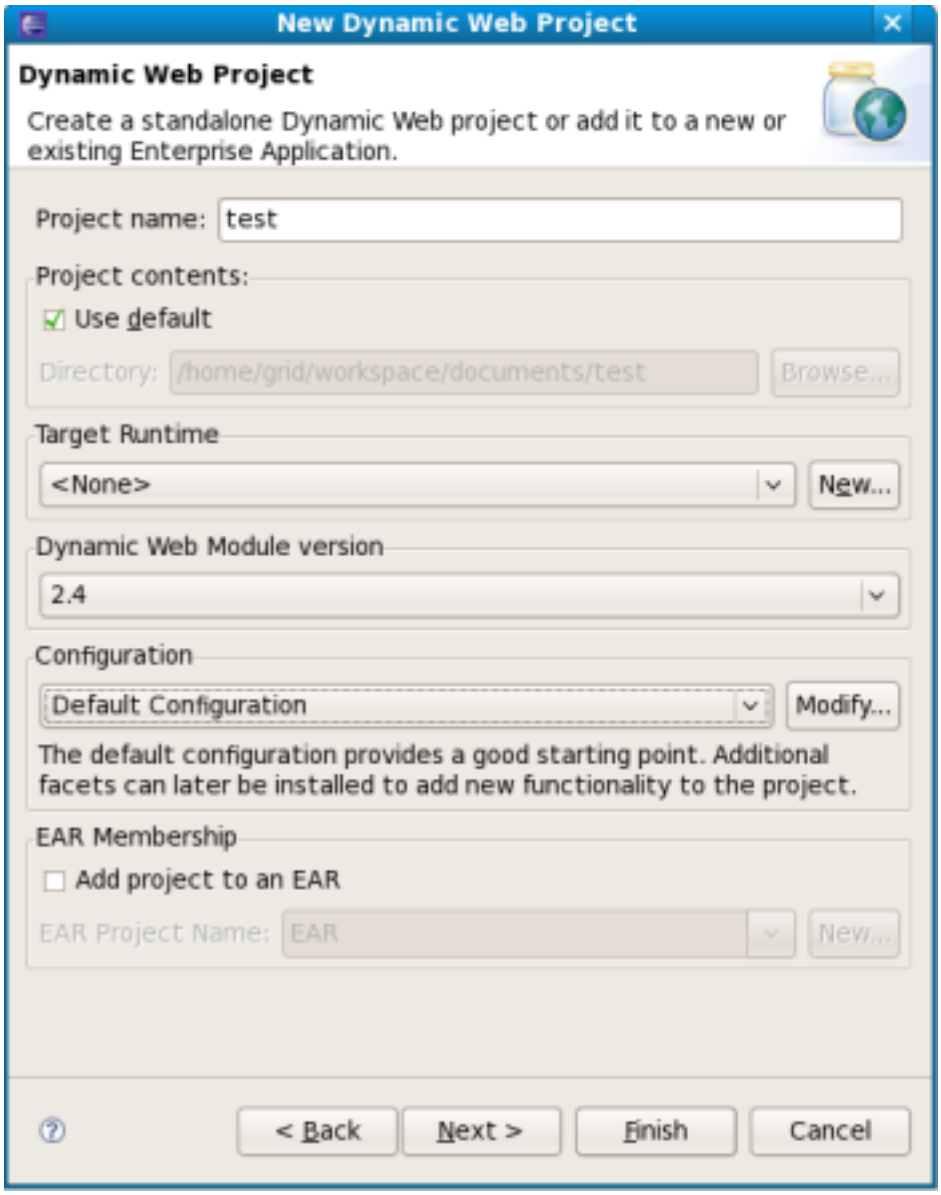

Figure 2.2. Dynamic Web Project Wizard

- Configuration: You may add *JBoss Web Service facet to the project* by clicking Modify... button. The opened page is like Figure 2.4.

- Configure Web Module values:

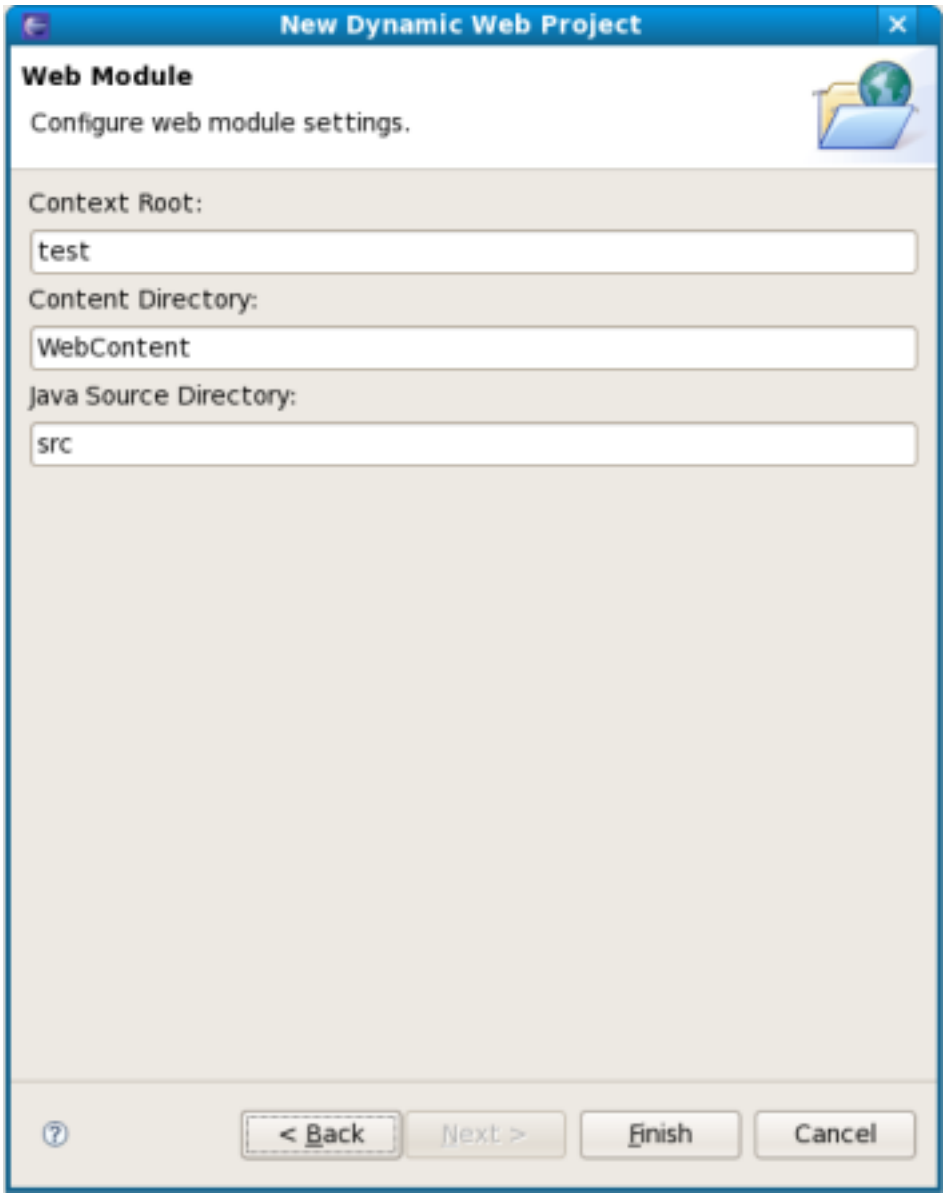

Figure 2.3. Web Module Settings Configuration

If you added the JBoss Web Service facet to the project, now the Finish button is disable. You must click Next button to set more information about the JBoss Web Service facet. The page is like Figure 2.5. Then click on the Finish button.

If you didn't add the JBoss Web Service facet to the project, click on the Finish button. Next you will need to add JBoss Web Service facet to the project.

2.2. Configure JBoss Web Service facet settings

If you have already created a new Dynamic Web project and not set the JBoss Web Service facet to the project, the next step is to add JBoss Web Service facet to the project. Right-click on the project, select its *Properties* and then find *Project Facets* in the tree-view on the left-side of the

project properties dialog. Tick on the check box for JBoss Web Services. You will see what like this:

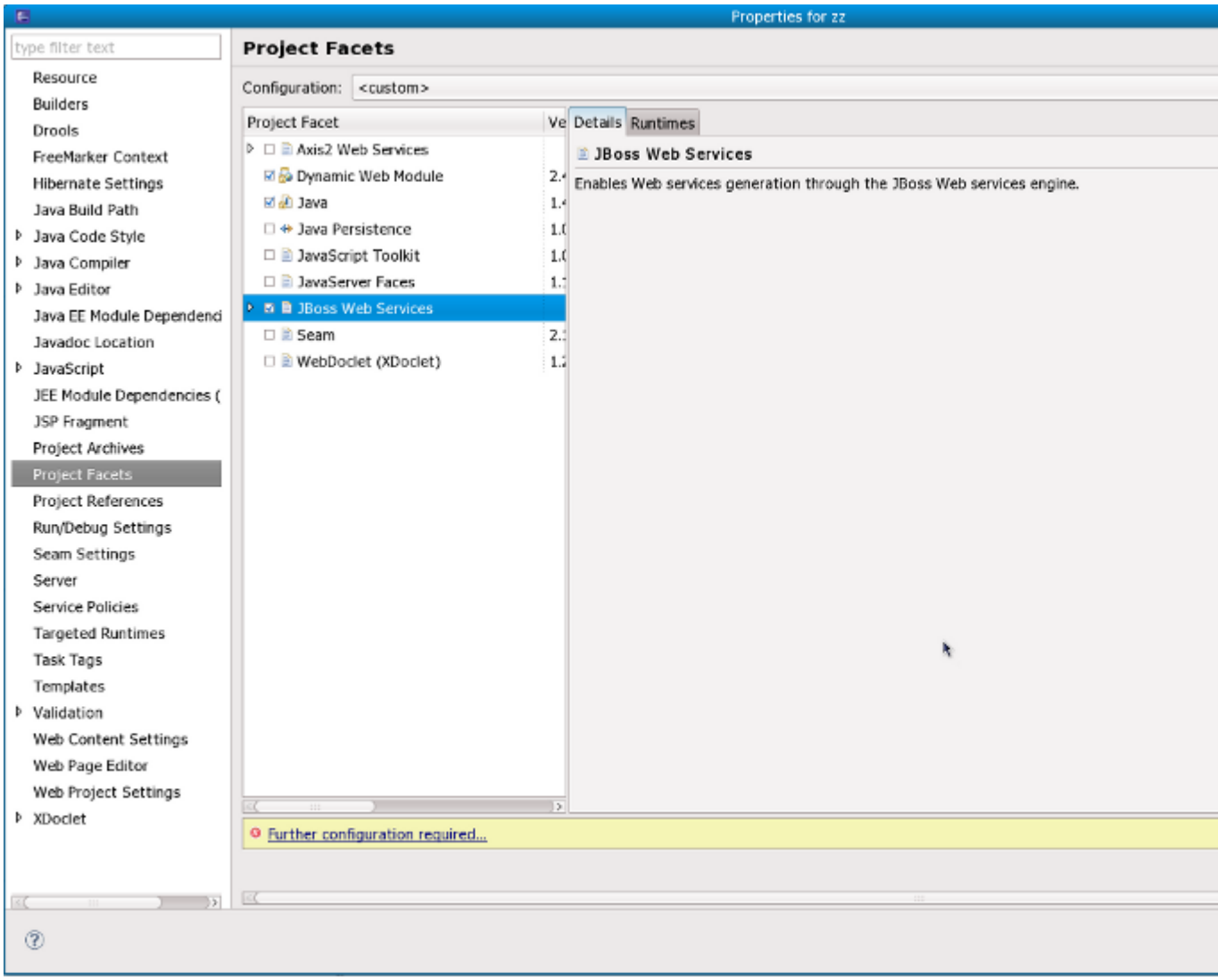

Figure 2.4. Choose JBoss Web Service Facet

At the bottom-left of the right-side of the project properties dialog, there is a error link: *Further configuration required...* . You must click the link to set more information about JBoss Web Service facet.

Click on the *Further configuration required...* link. In the opened window

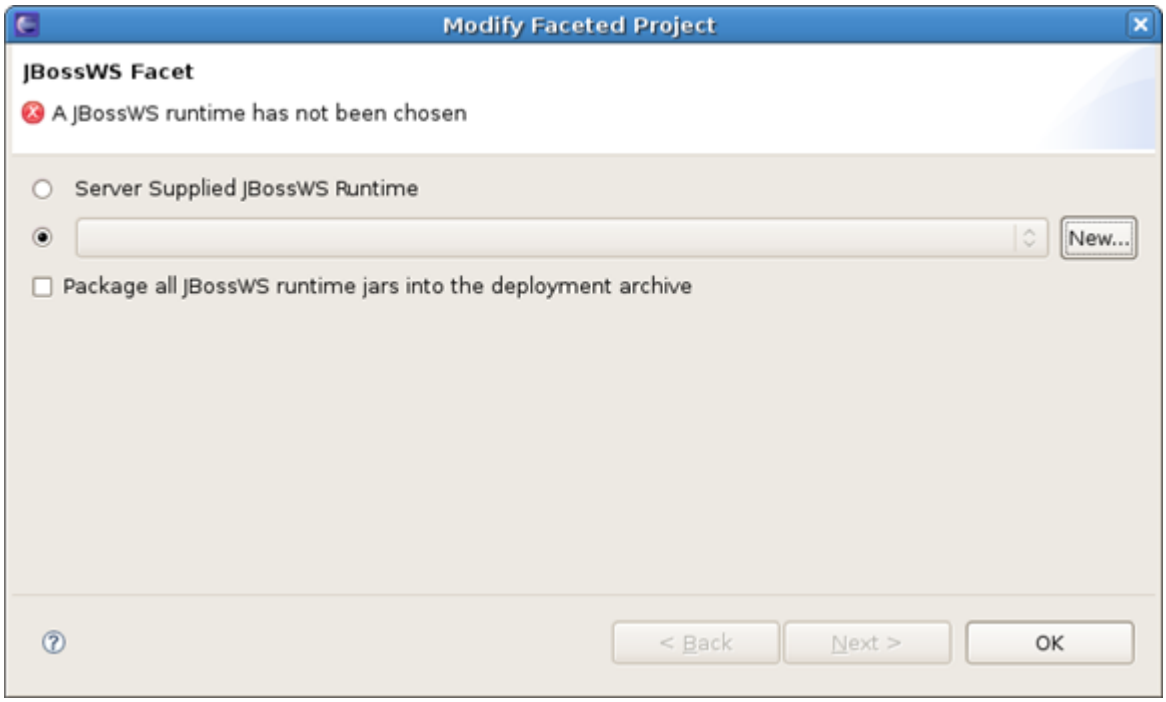

Figure 2.5. Configure JBoss Web Service Facet

Server Supplied JBossWS Runtime: If you have already set a JBoss runtime to the project's target runtime, you may choose *Server Supplied JBossWS Runtime* and then click *Ok* to finish the configuration of JBoss Web Service facet.

If the project has no *Target Runtime* settings, you should check the second radio button and specify a JBossWS runtime from the list. You also can create a new JBossWS runtime, click on the *New...* button will bring you to another dialog to configure new JBossWS runtime.

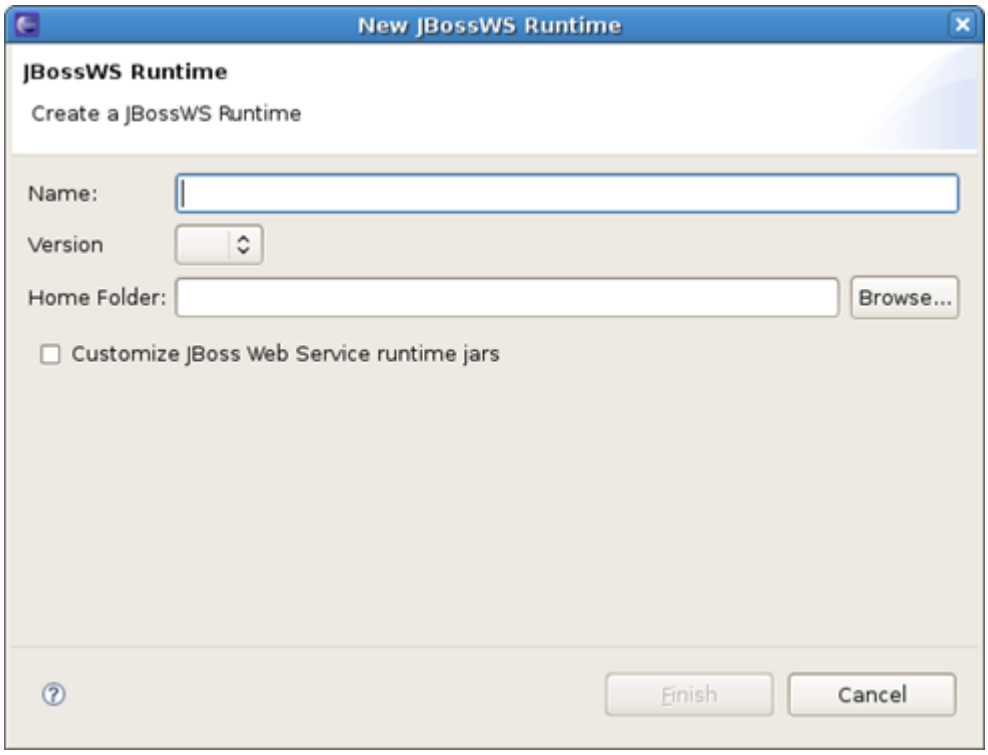

Figure 2.6. Configure JBossWS Runtime

See how to configure a new JBossWS runtime *here*.

After setting the information about JBoss Web Service facet, for saving the result, you should click the Apply or OK button at the bottom-right of the right-side of the project properties dialog.

2.3. Creating a Web Service from a WSDL document using JBossWS runtime

In this chapter we provide you with the necessary steps to create a Web Service from a WSDL document using JBossWS runtime.

At first, please make sure that you have already created a dynamic Web project with JBoss Web Service facet installed.

See how to make it *here* and *here*.

To create a Web Service using JBossWS runtime select *File > New > Other > Web Services > Web Service* to run Web Service creation wizard.

Let's get through the wizard step-by-step:

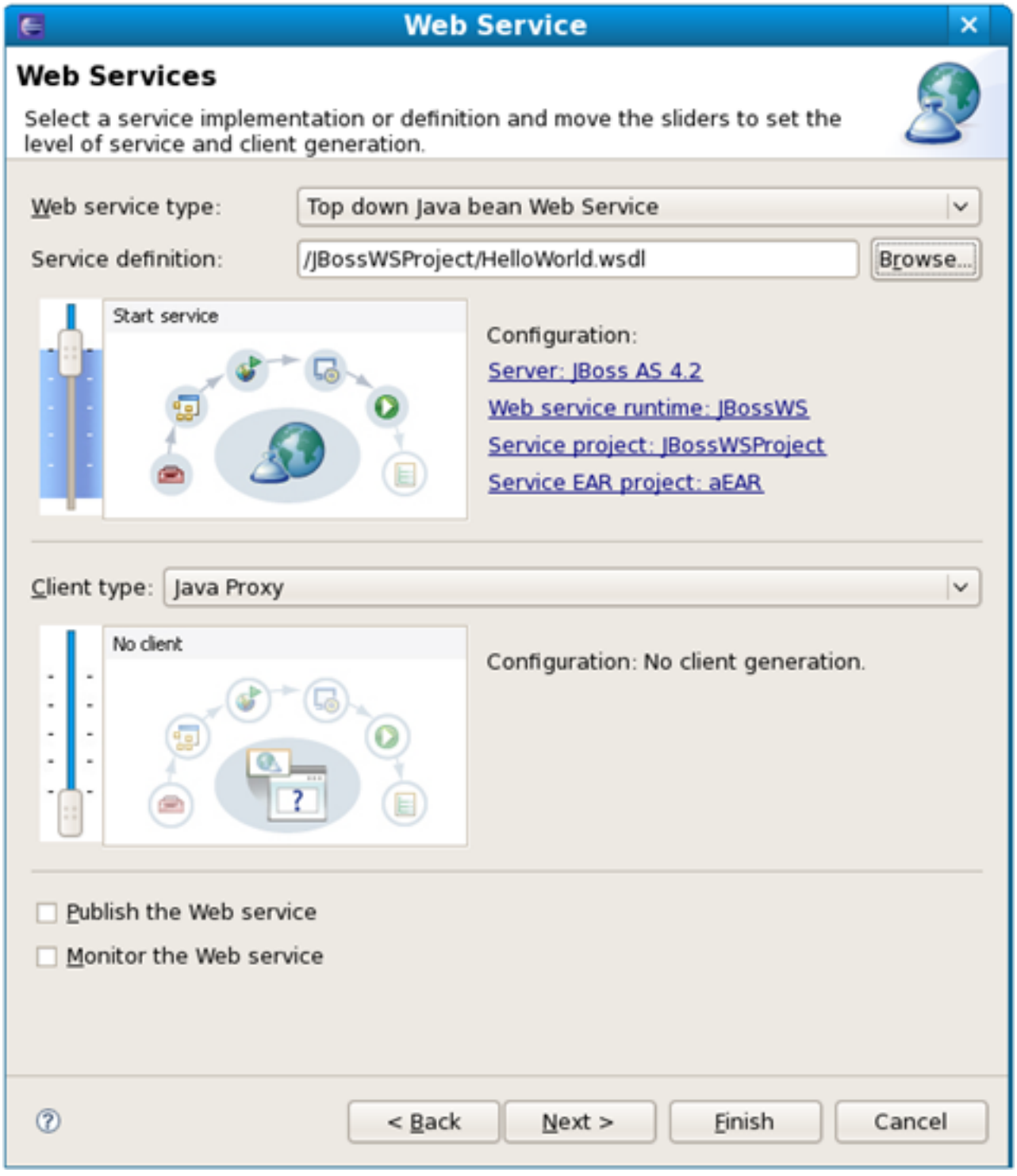

Figure 2.7. New Web Service Wizard

First, please select Top down Java bean Web Service from the Web Service type list, and select a WSDL document from workspace, click on the Server name link on the page will bring you to another dialog. Here you can specify the server to a JBoss Server and Web Service runtime to JBossWS runtime:

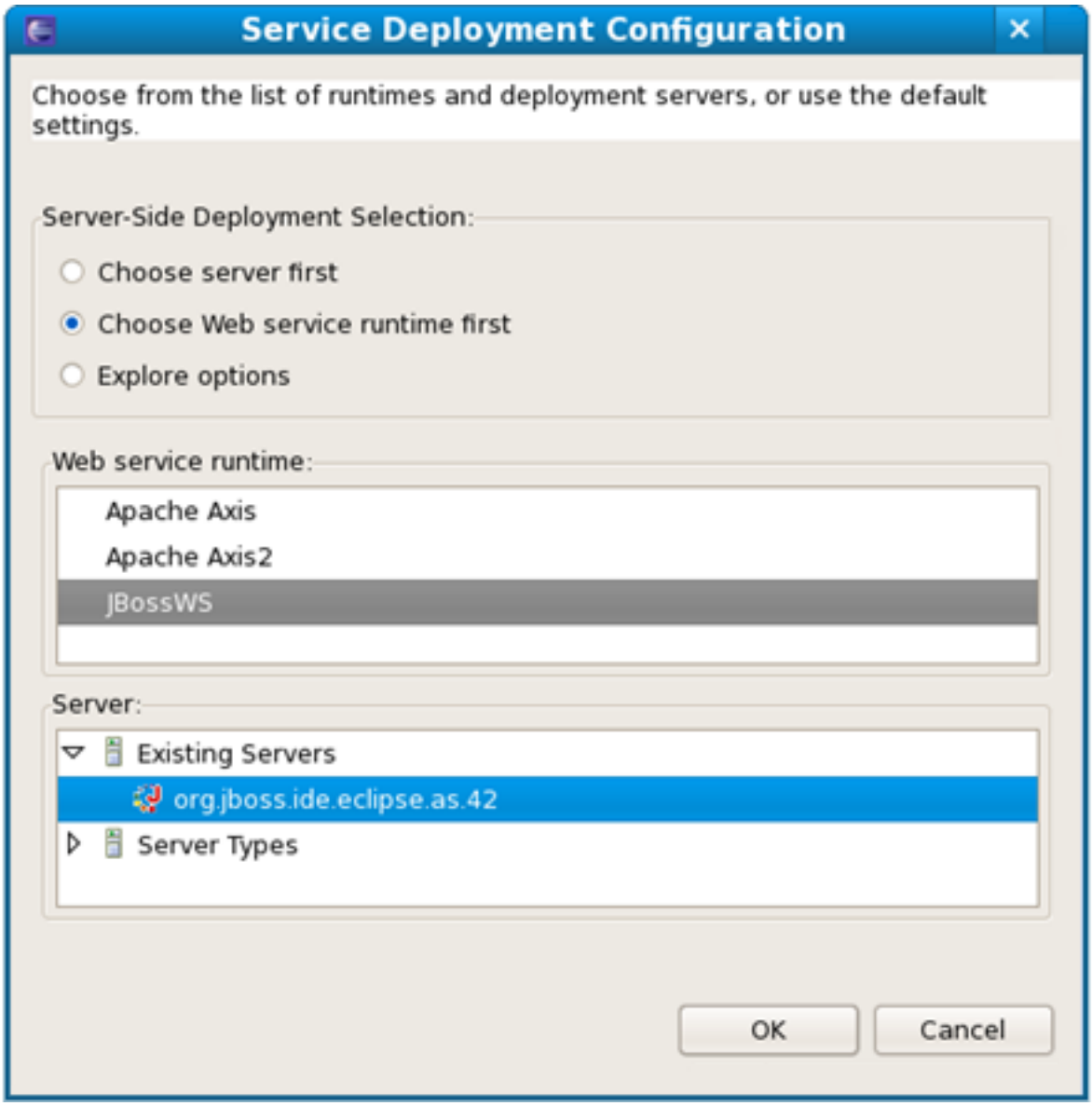

Figure 2.8. Select Server and Web Service runtime

Click on the *Finish* button to see the next wizard view opened:

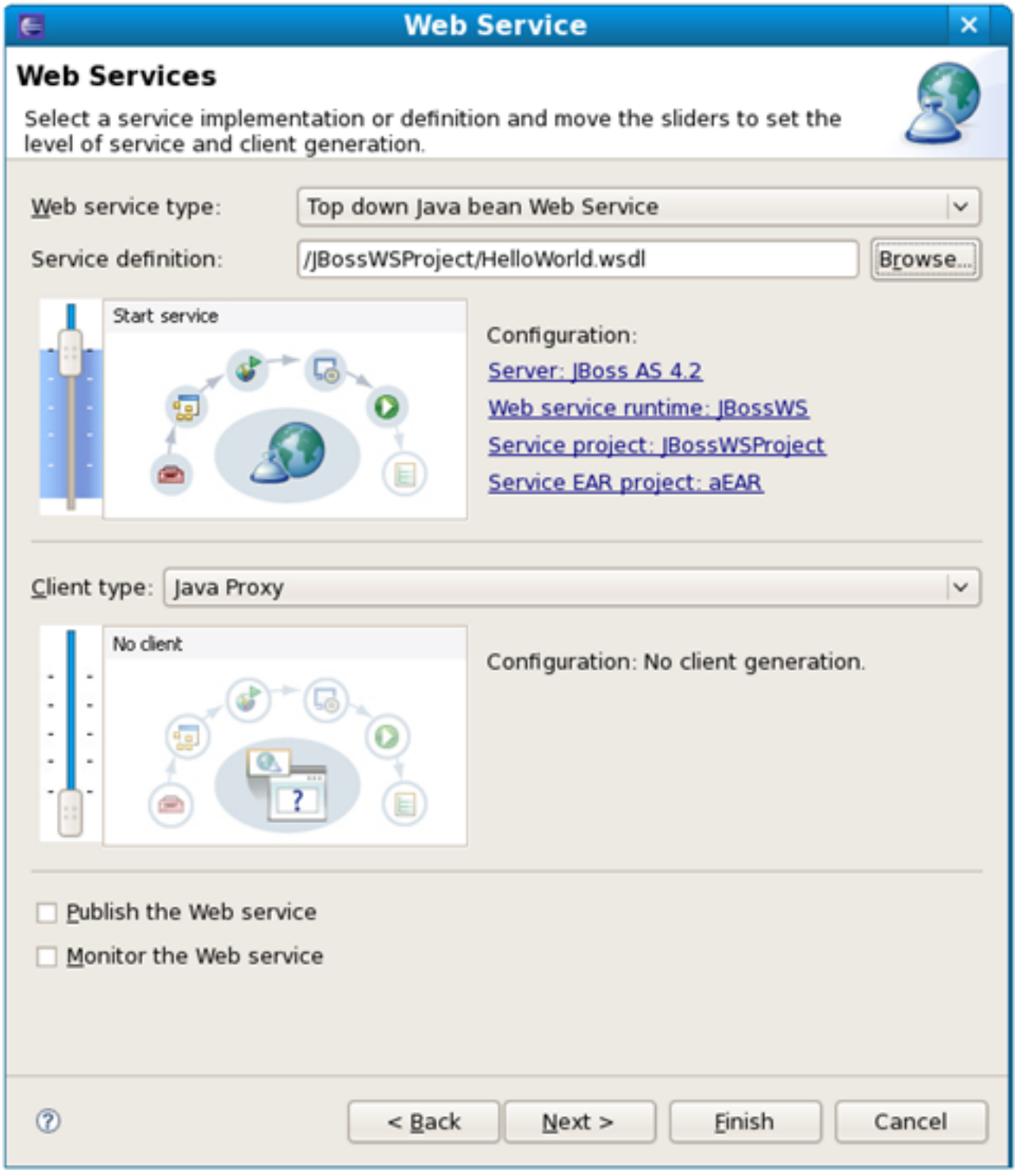

Figure 2.9. New Web Service Wizard

Click on the *Next* button to proceed:

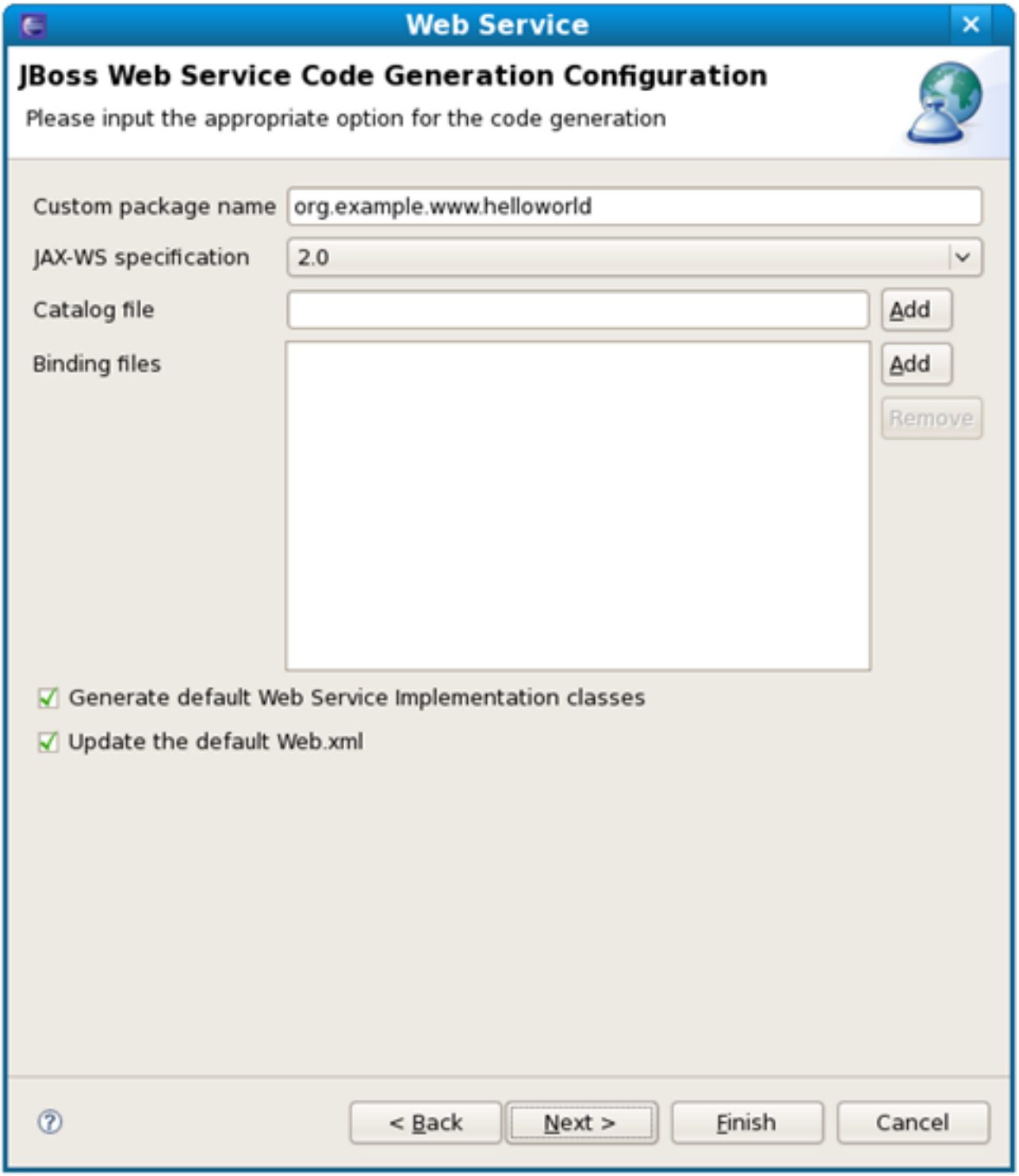

Figure 2.10. New Web Service Wizard

On this page, the default package name comes from the namespace of the WSDL document, you also can change it to any valid package name you want. JAX-WS specification should be set to 2.0 if your JBossWS runtime in JBoss Server is JBossWS native runtime. You can specify a catalog file and binding files if you have them. If you want the wizard to generate empty implementation

classes for the Web Service, check the *Generate default Web Service implementation classes* check box. If you want to update the default Web.xml file with the Web Service servlets configured, check the *Update the default Web.xml* check box. Click on the *Next* or on the *Finish* button to generate code.

Once the Web Service code is generated, you can view the implementation class and add business logic to each method.

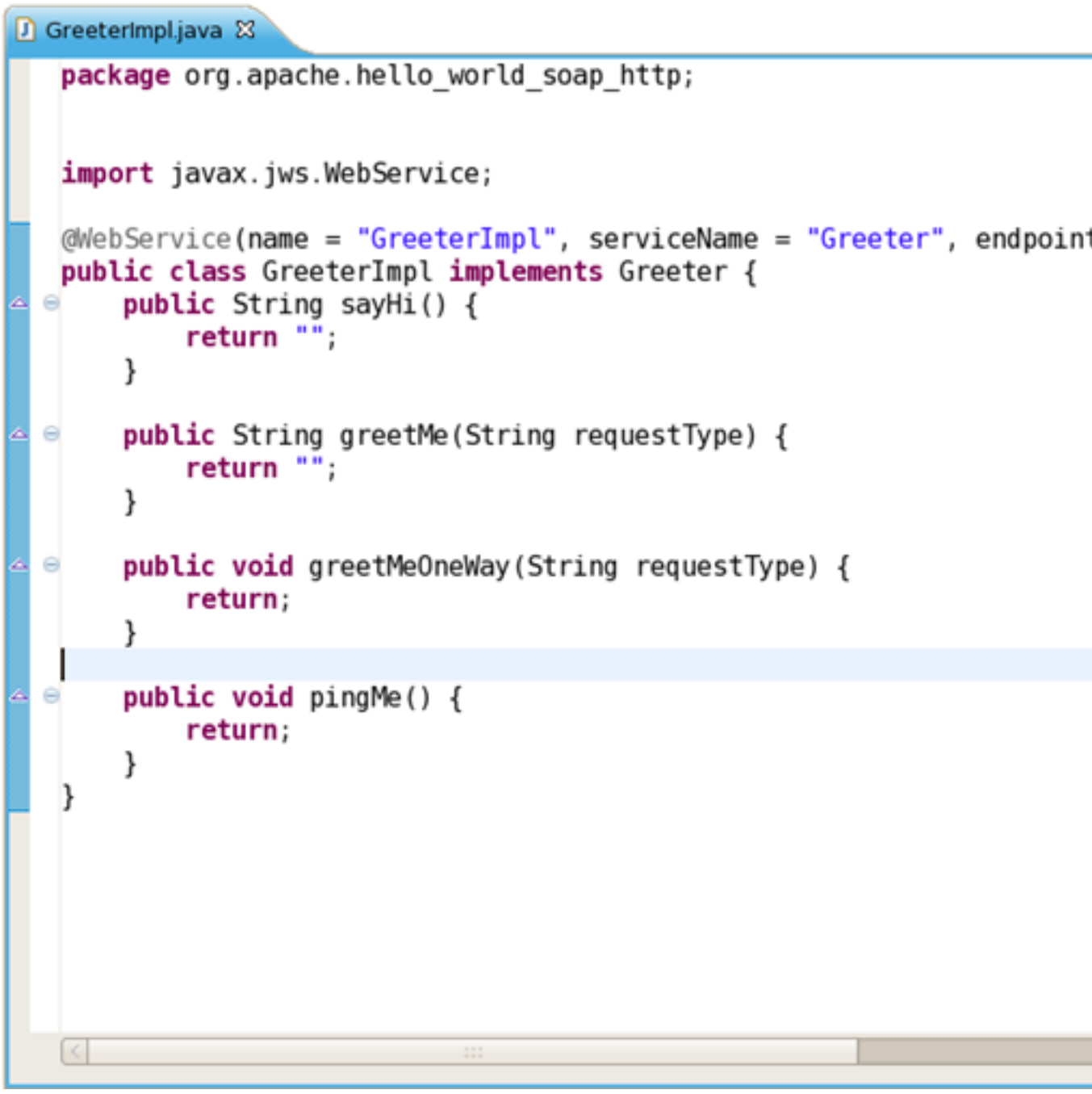

Figure 2.11. The generated implementation Java code

View the Web.xml file:

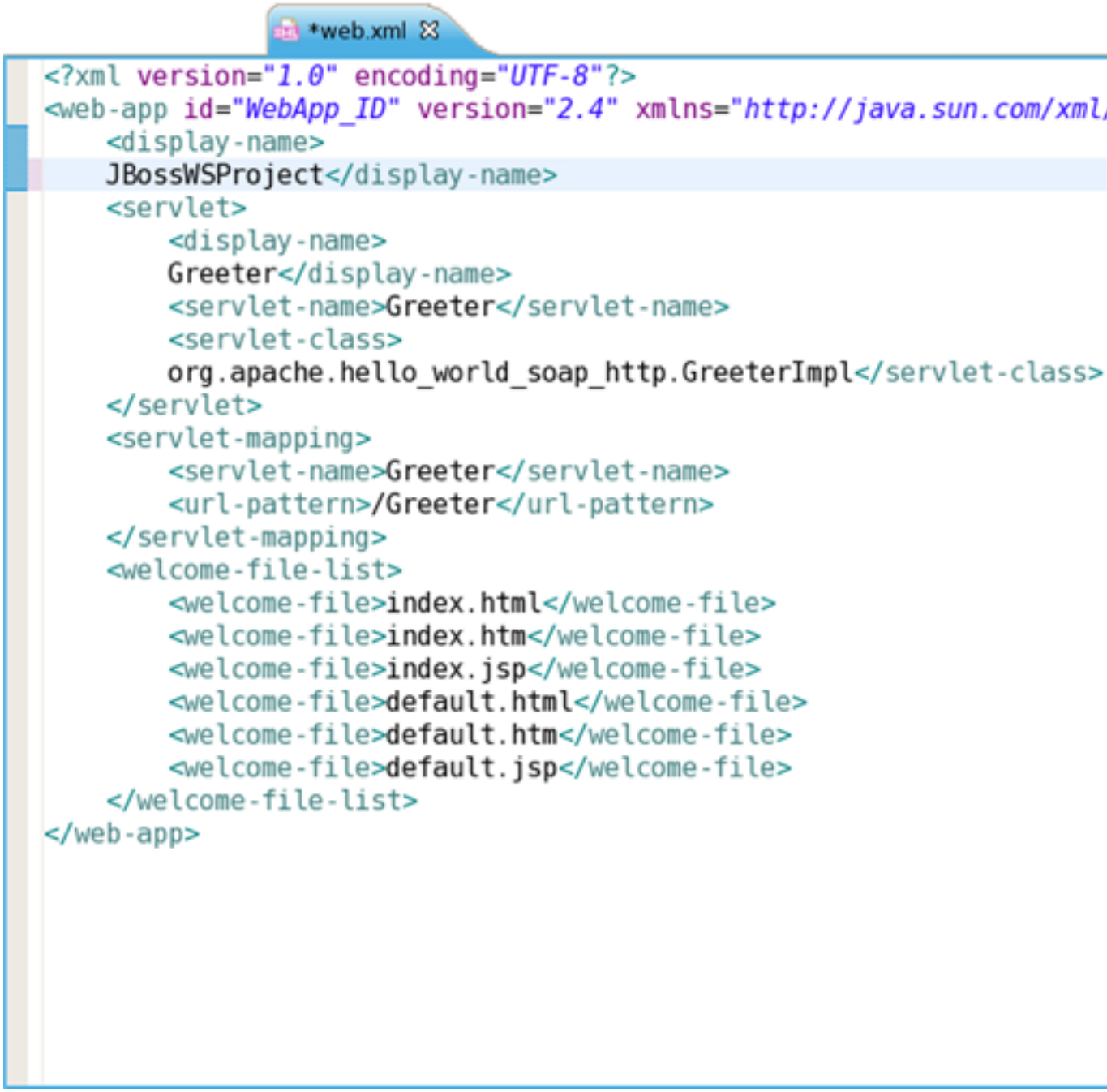

Figure 2.12. Web.xml

In the next chapter you will find out how to create a Web service from a Java bean.

2.4. Creating a Web service from a Java bean using JBossWS runtime

The Web Service wizard assists you in creating a new Web service, configuring it for deployment, and then deploying it to the server.

To create a Web service from a bean using JBoss WS:

Setup *JBoss WS and development environment*.

Create *a Dynamic Web project*.

Add *JBossWS Facet* to Web project.

Create a Web Service from a java bean:

- Switch to the Java EE perspective *Window > Open Perspective > Java EE*.
- In the Project Explorer view, select the bean that you created or imported into the source folder of your Web project.

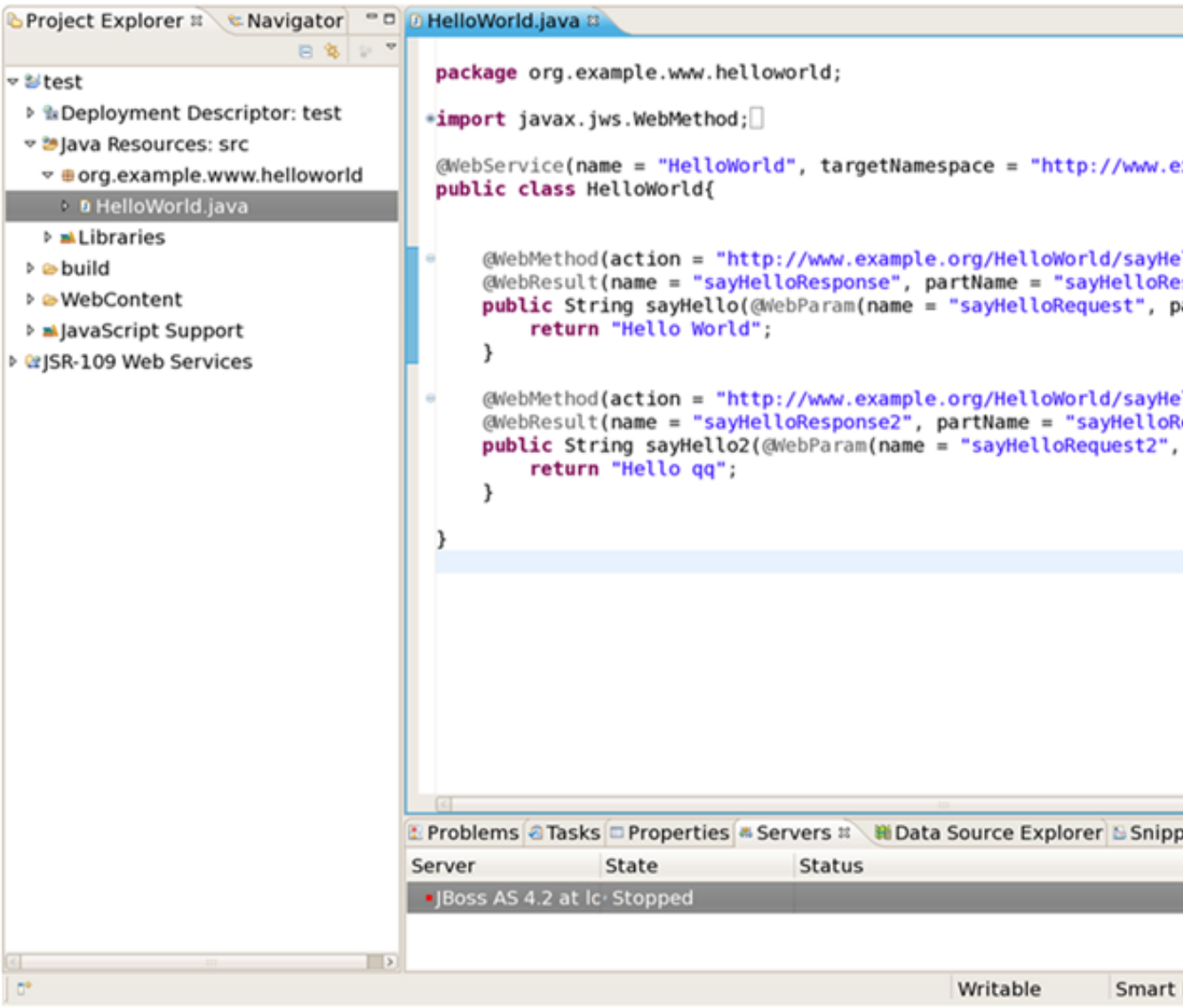

Figure 2.13. Select the Bean Created

- Click *File > New > Other*. Select Web Services in order to display various Web service wizards. Select the Web Service wizard. Click on the Next button.

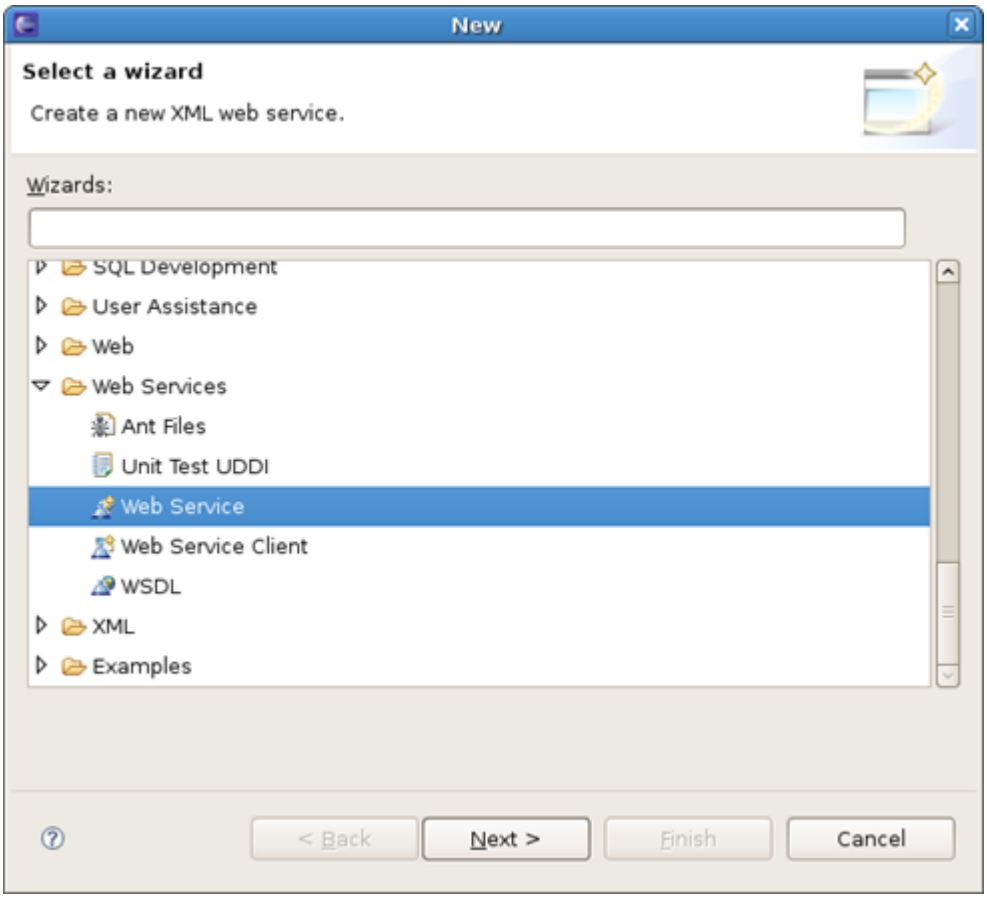

Figure 2.14. New Web Service

- On the first Web Service wizard page: select Bottom up Java bean Web service as your Web service type, and select the Java bean from which the service will be created:

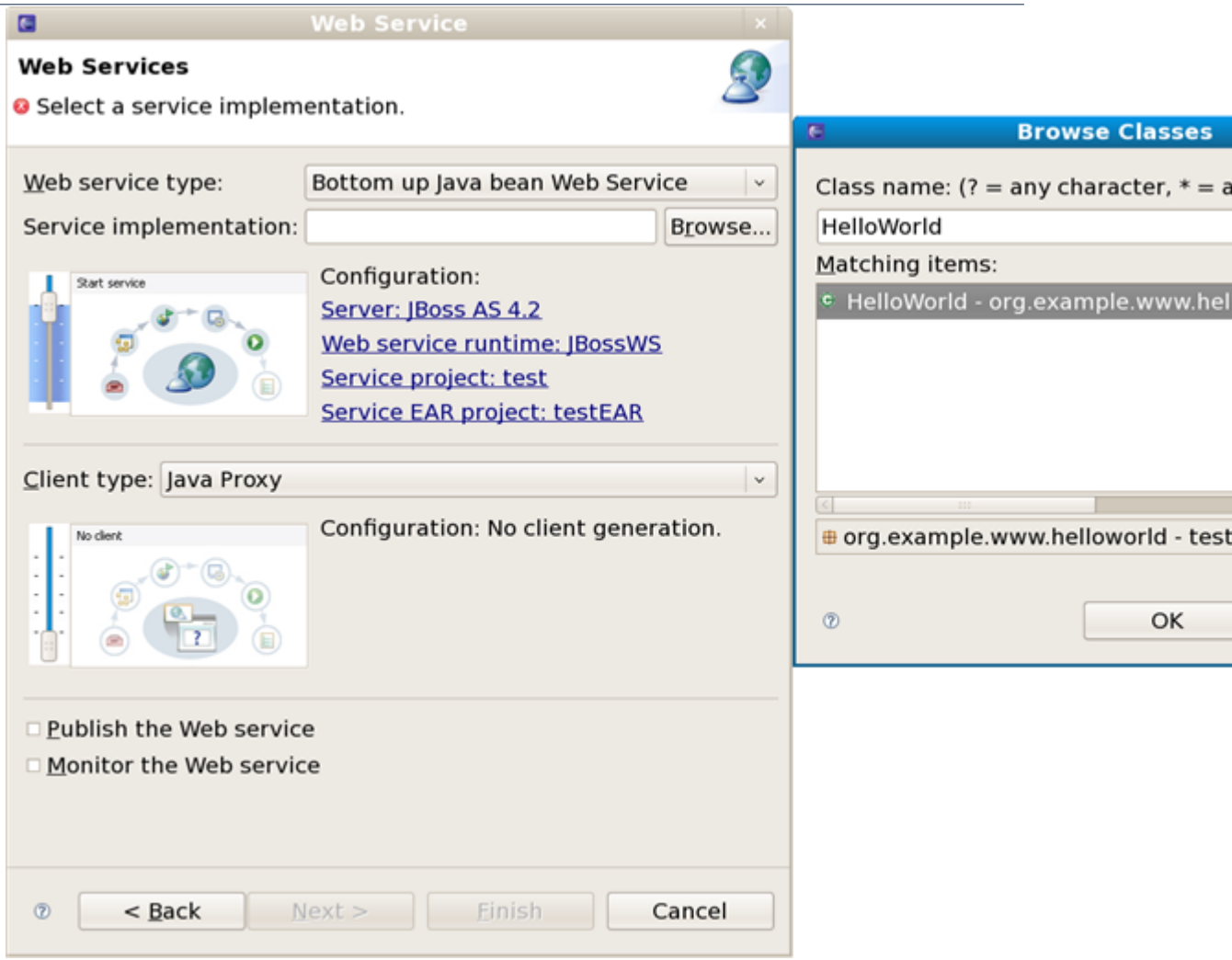

Figure 2.15. Set Web Service Common values

- Select the stages of Web service development that you want to complete using the slider:
  - Develop: this will develop the WSDL definition and implementation of the Web service. This includes such tasks as creating modules that will contain generated code, WSDL files, deployment descriptors, and Java files when appropriate.
  - Assemble: this ensures the project that will host the Web service or client gets associated to an EAR when required by the target application server.
  - Deploy: this will create the deployment code for the service.
  - Install: this will install and configure the Web module and EARs on the target server.
  - Start: this will start the server once the service has been installed on it. The server-config.wsdd file will be generated.

    - Test: this will provide various options for testing the service, such as using the Web Service Explorer or sample JSPs.

  - Select your server: the default server is displayed. If you want to deploy your service to a different server click the link to specify a different server.

  - Select your runtime: ensure the JBoss WS runtime is selected.

  - Select the service project: the project selected in your workspace is displayed. To select a different project click on the project link. If you are deploying to JBoss Application Server you will also be asked to select the EAR associated with the project. Ensure that the project selected as the Client Web Project is different from the Service Web Project, or the service will be overwritten by the client's generated artifacts.

  - If you want to create a client, select the type of proxy to be generated and repeat the above steps for the client. The better way is to create a web service client project separately.

  Click on the Next button.

- On the JBoss Web Service Code Generation Configuration page, set the following values:

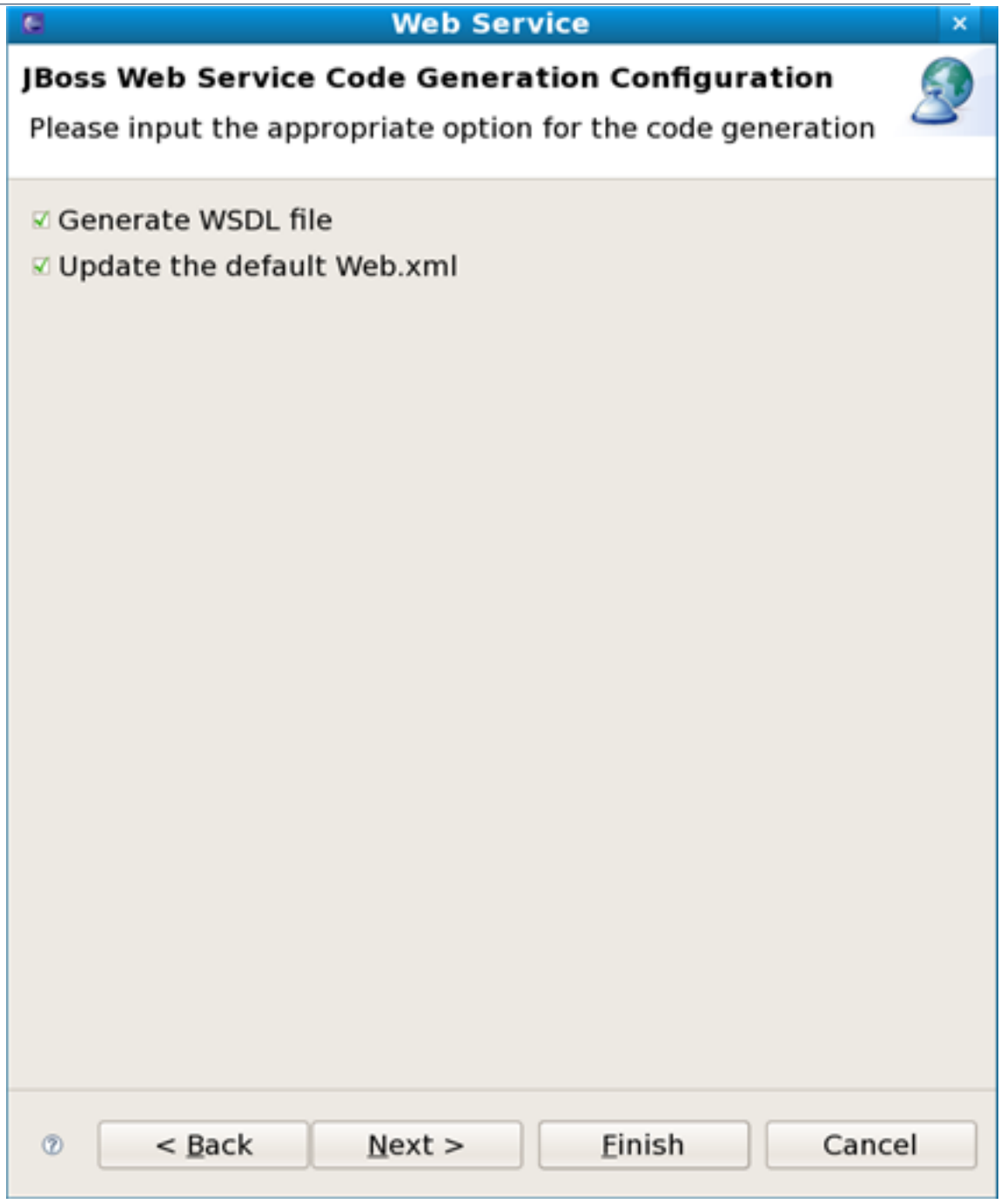

Figure 2.16. Set Web Service values for Code Generation

- Generate WSDL file: select it, you will get a generated WSDL file in your project. But this wsdl's services' address location values are not a real address.
- After the Web service has been created, the following option can become available depending on the options you selected: Update the default web.xm file. If selected, you may test the web service by Explorer.

Click on the Next button.

- On this page, the project is deployed to the server. You can start the server and test the web service. If you want to publish the web service to a UDDI registry, you may click the Next button to publish it. If not, you may click the Finish button.

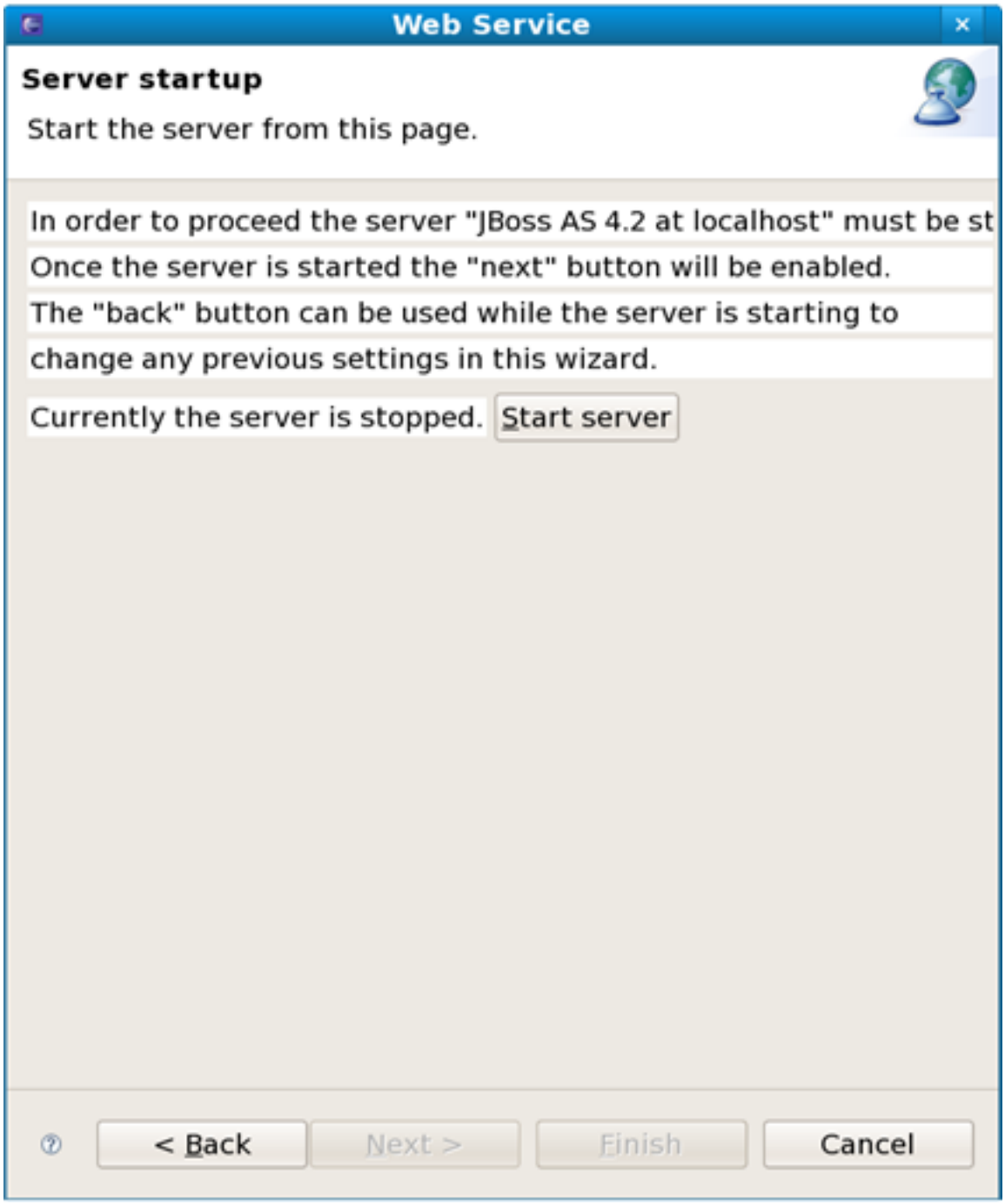

Figure 2.17. Start a Server

After the Web Service has been created, the following options may become available depending on the options selected:

- the generated web services code
- If you selected to generate a WSDL file, you will get the file in your project's wsdl folder.

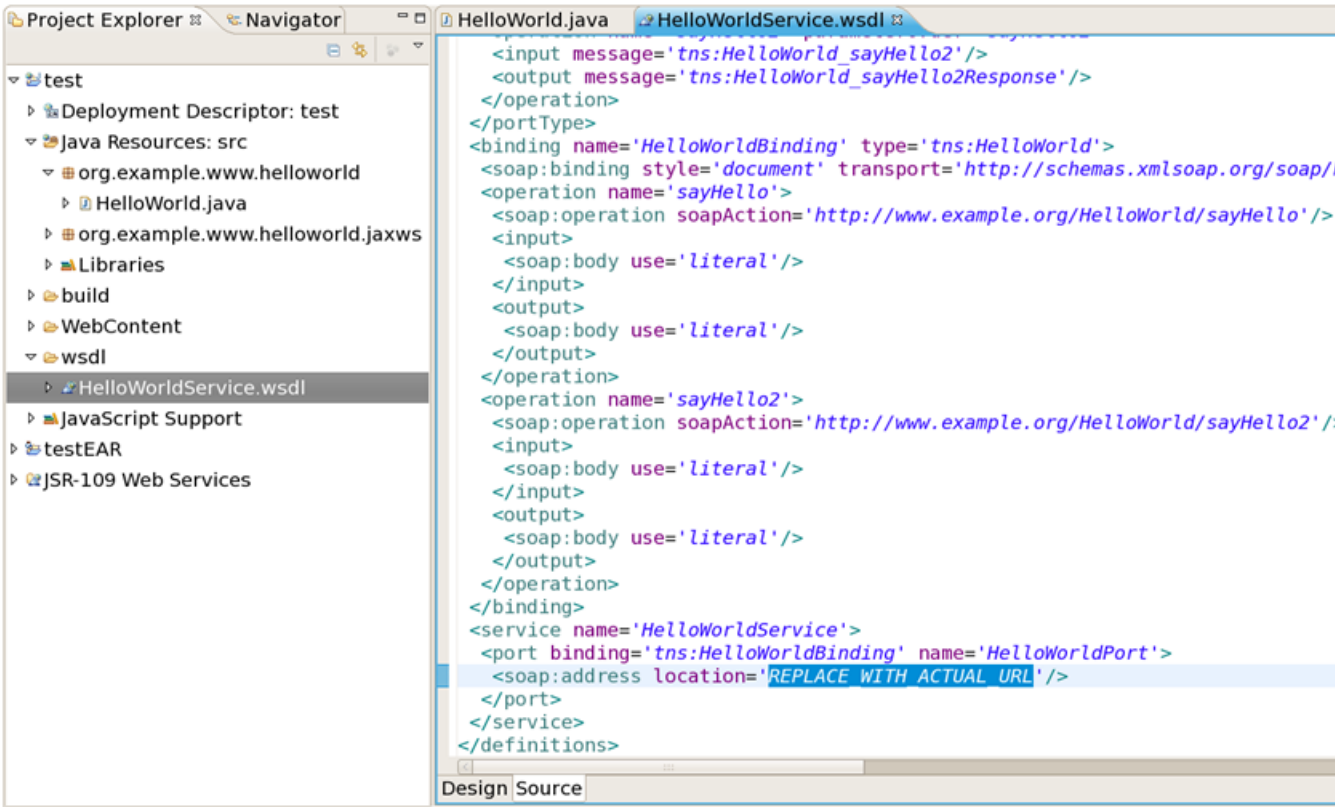

Figure 2.18. The Generated HelloWorldService.wsdl File in the wsdl Folder

- If you selected to update the default web.xml, you will test the web service in the browser. Open the Explorer, input the url for the web service according to web.xml plus ?wsdl., you will get the WSDL file from Explorer.

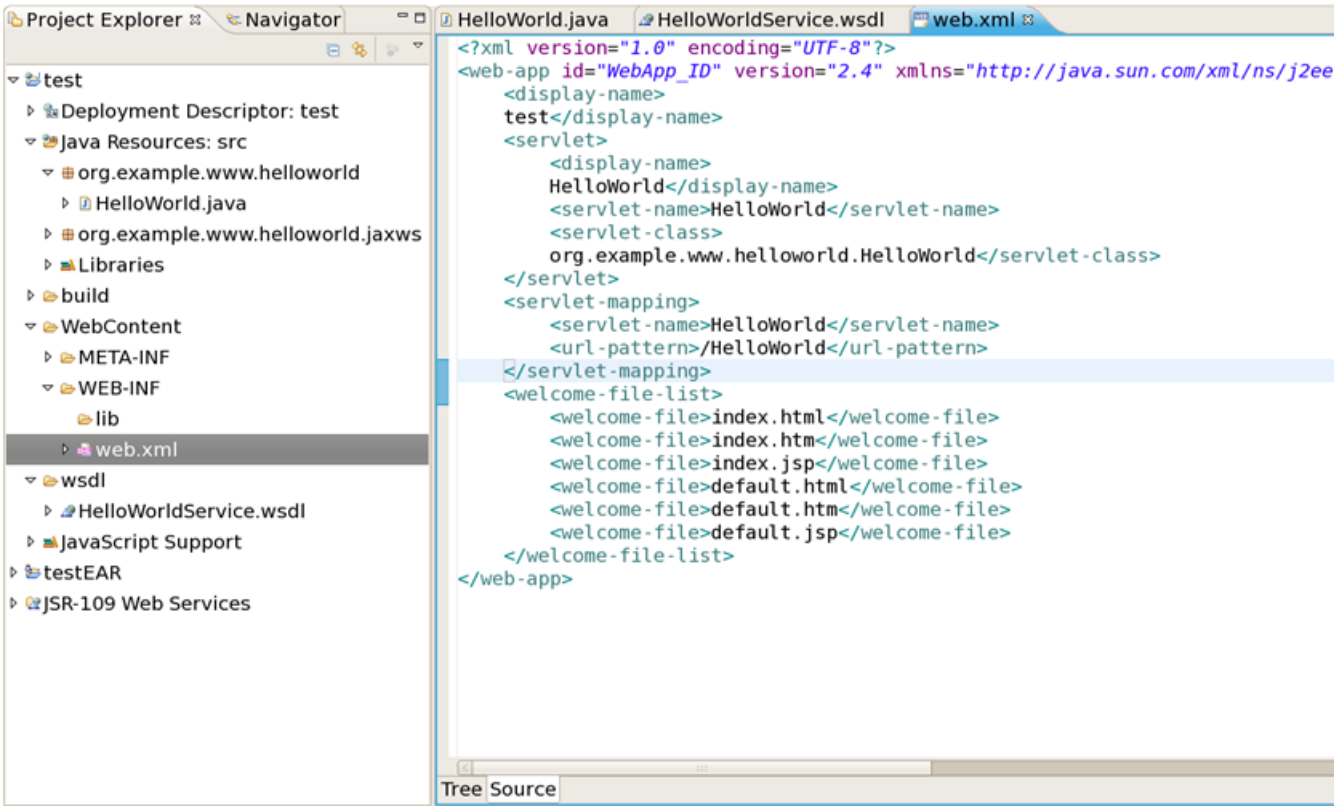

Figure 2.19. The Updated web.xml file

In the next chapter you will be able to create a Web Service Client from a WSDL document using JBoss WS.

Creating a Web Service Client from a WSDL Document using JBoss WS

To create a Web Service Client from a WSDL Document using JBoss WS you need to fulfil the following steps:

Setup *JBoss WS and development environment*.

Create a Dynamic Web project.

Add JBossWS Facet to Web project.

Then you can create a Web Service Client from a WSDL document:

- Switch to the Java EE perspective *Window > Open Perspective > Java EE*.
- Click *File > New > Other*. Select Web Services in order to display the various Web service wizards. Select the Web Service Client wizard. Click on the Next button.

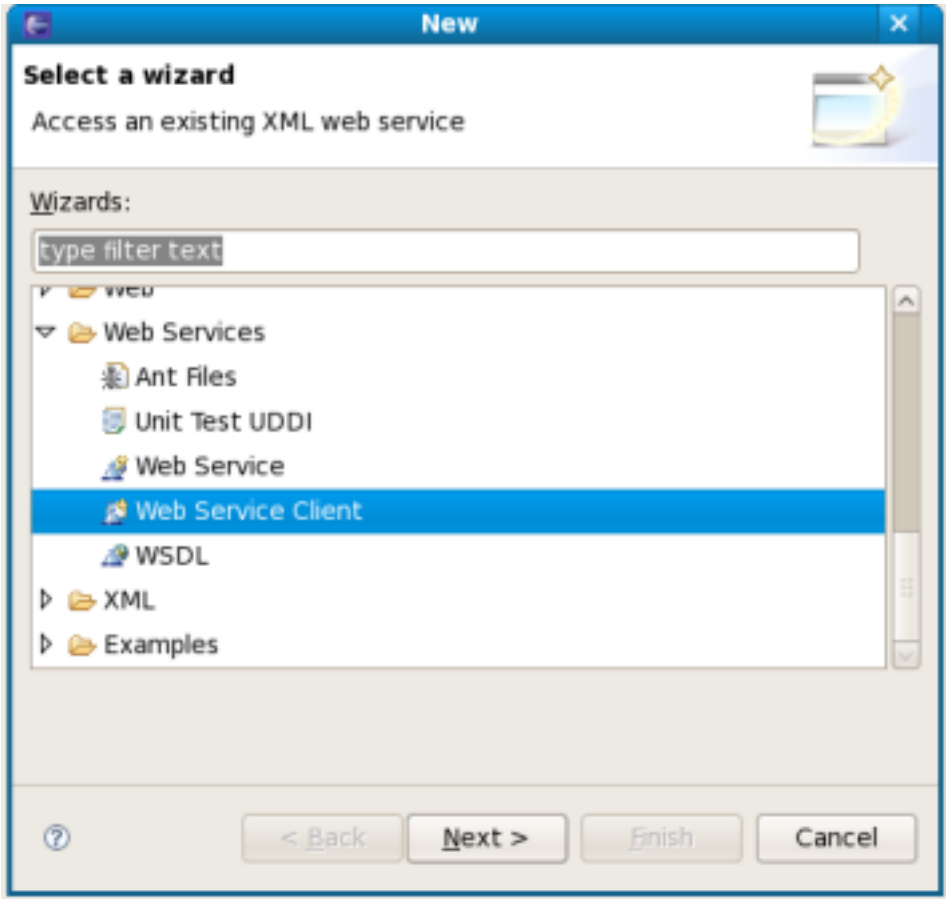

Figure 3.1. New Web Service Client

- The first and the second Web Service Client wizard pages are the same as for *Web Service from a WSDL document*.

Figure 3.2. Set Web Service Common values

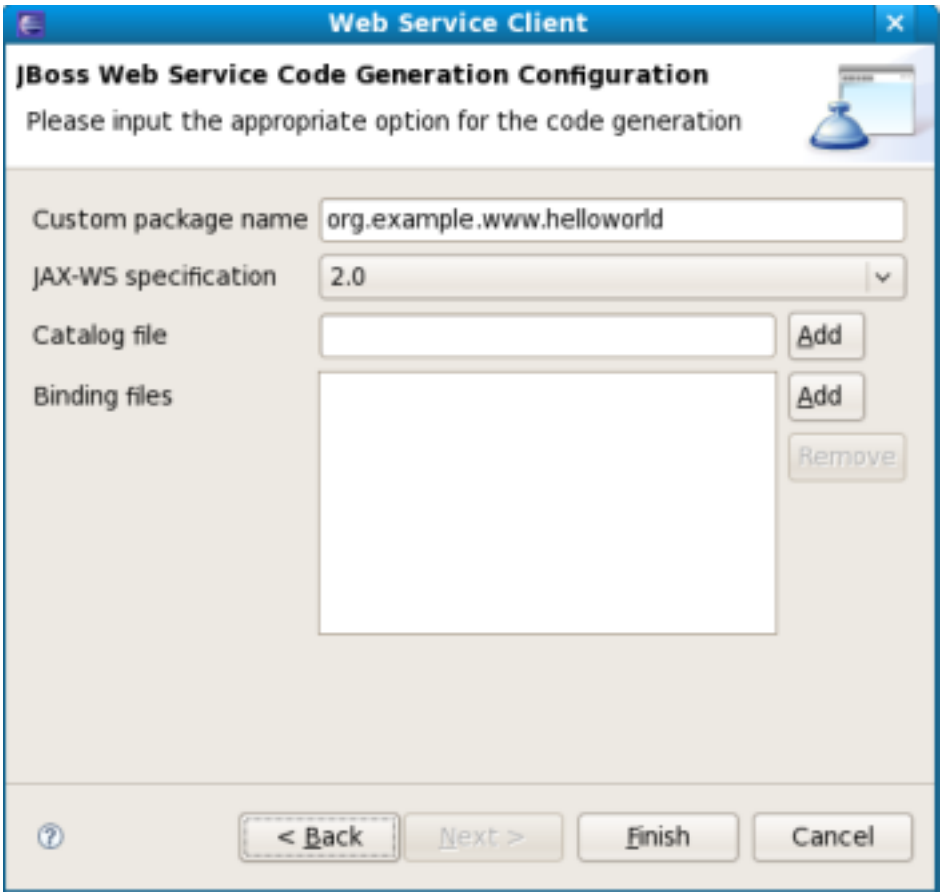

Figure 3.3. Set Web Service values related to WSDL file

The only difference is:

- Client Type: Support of Java Proxy only.

Click on the Finish button.

After the Web Service Client has been created, the following may occur depending on the options you selected:

- the generated web service and client codes
- a client sample class.

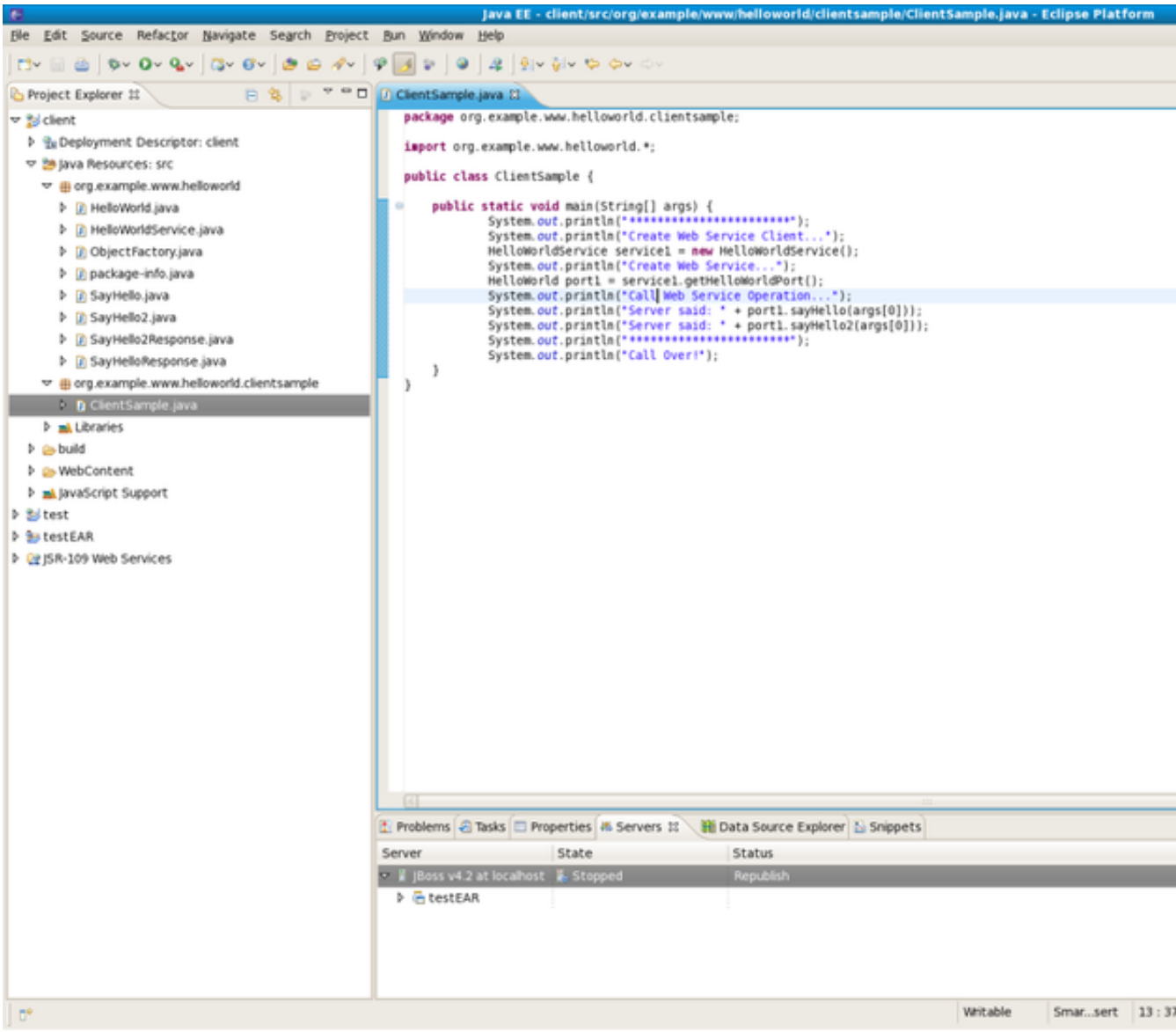

Figure 3.4. Client Sample Class

JBoss WS use a Java class to test Web Service. A client sample class will be generated, you may run this client as a java application to call a web service.

JBoss WS and development environment

In this chapter you will learn how to change JBossWS preferences and how to set default server and runtime.

4.1. JBossWS Preferences

In this section you will know how JBossWS preferences can be modified during the development process.

JBossWS preferences can be set on the JBossWS preference page. Click on *Window > Preferences > JBoss Tools > Web > JBossWS Preferences*.

On this page you can manage the JBossWS Runtime. Use the appropriate buttons to Add more runtimes or to Remove those that are not needed.

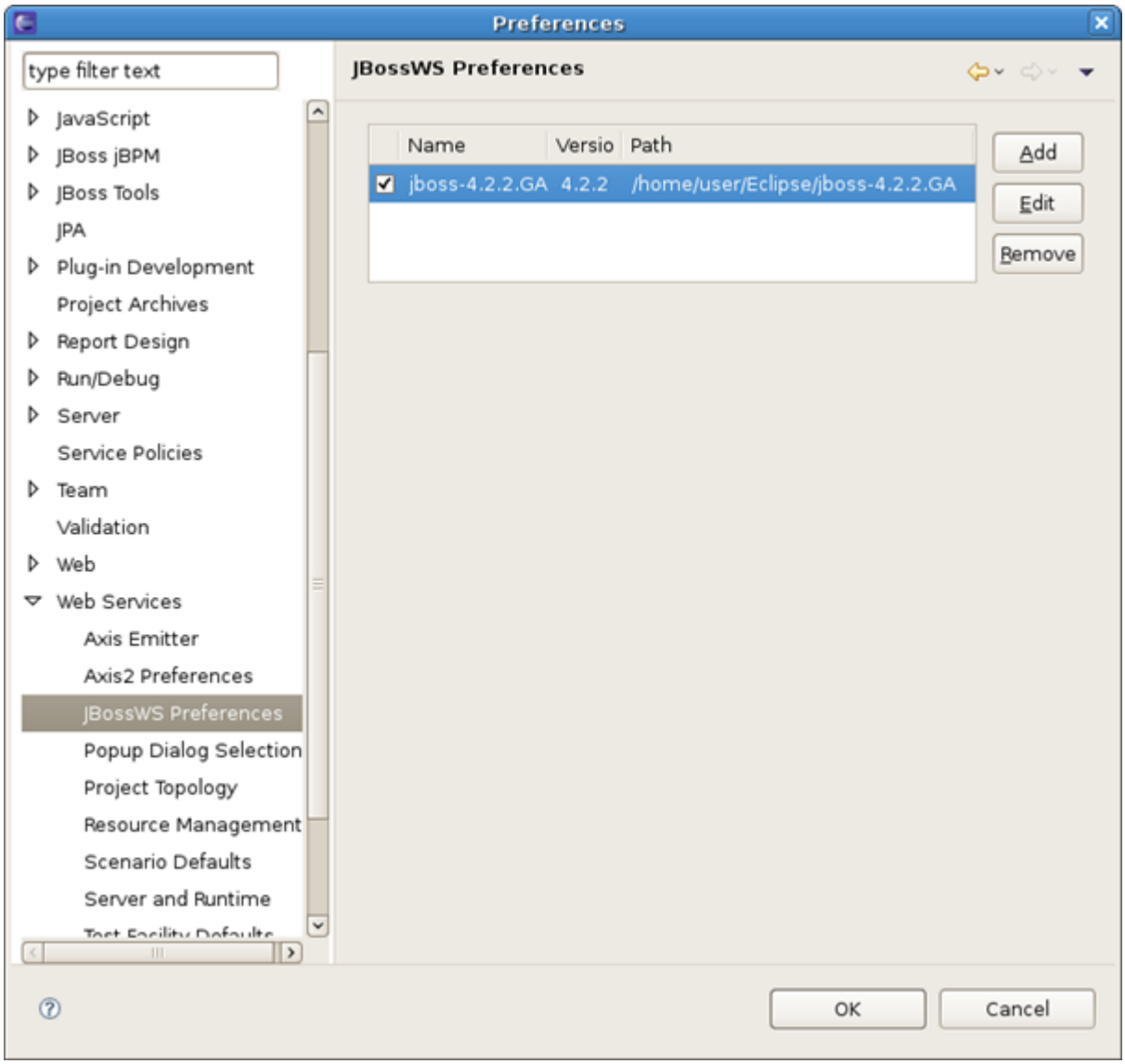

Figure 4.1. JBossWS Preferences Page

Clicking on *Add* or *Edit* button will open the form where you can configure a new JBossWS runtime and change the path to JBossWS runtime home folder, modify the name and version of the existing JBossWS runtime settings. Press Finish to apply the changes.

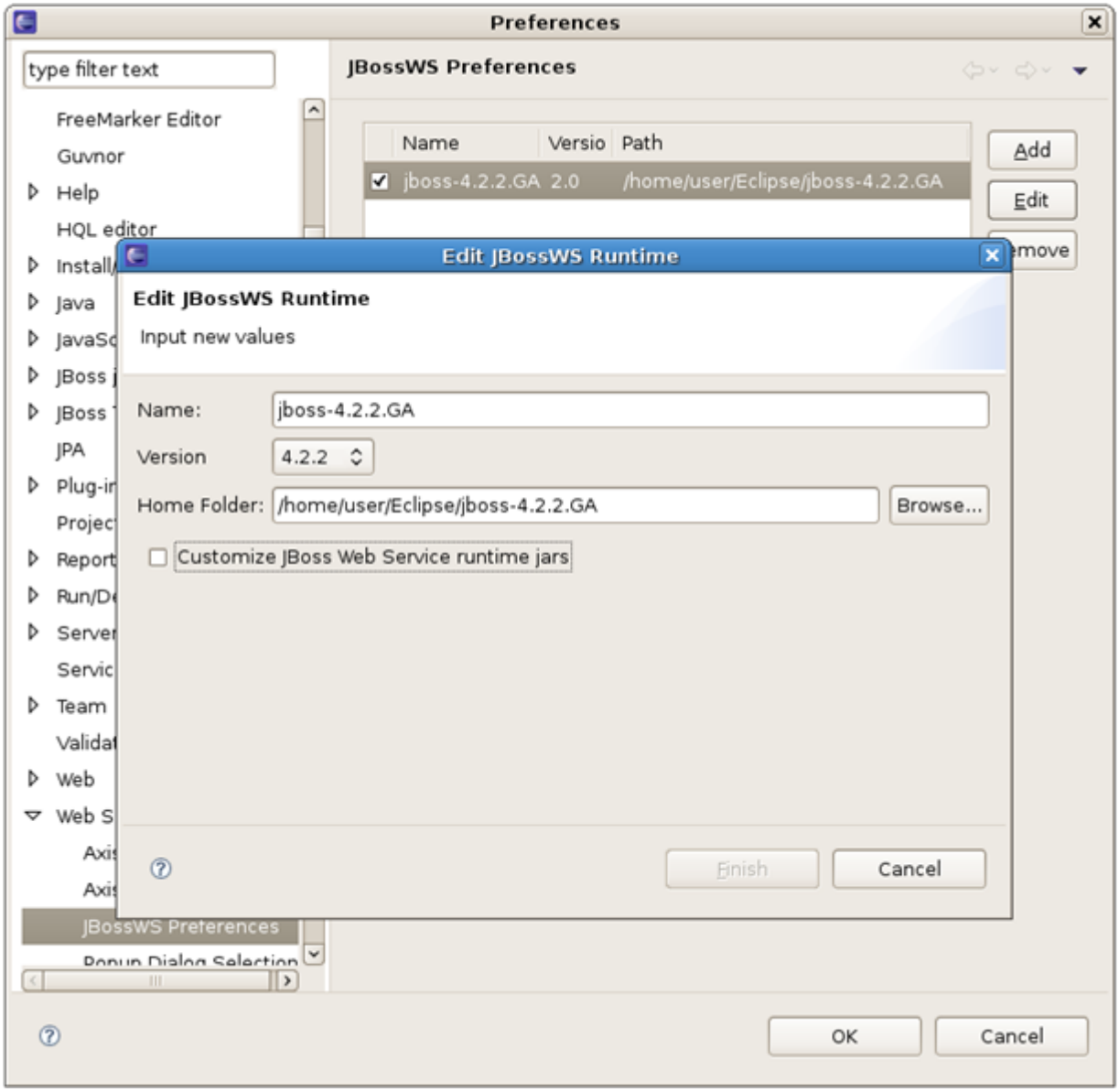

Figure 4.2. Edit JBossWS Runtime

WS container allows Source and JavaDoc locations to be set via the Properties dialog on each contained .jar: right-click on any .jar file in the Project Explorer view, select *Properties*. Choose *Java Source Attachment* and select location (folder, JAR or zip) containing new source for the chosen .jar using one of the suggested options (workspace, external folder or file) or enter the path manually:

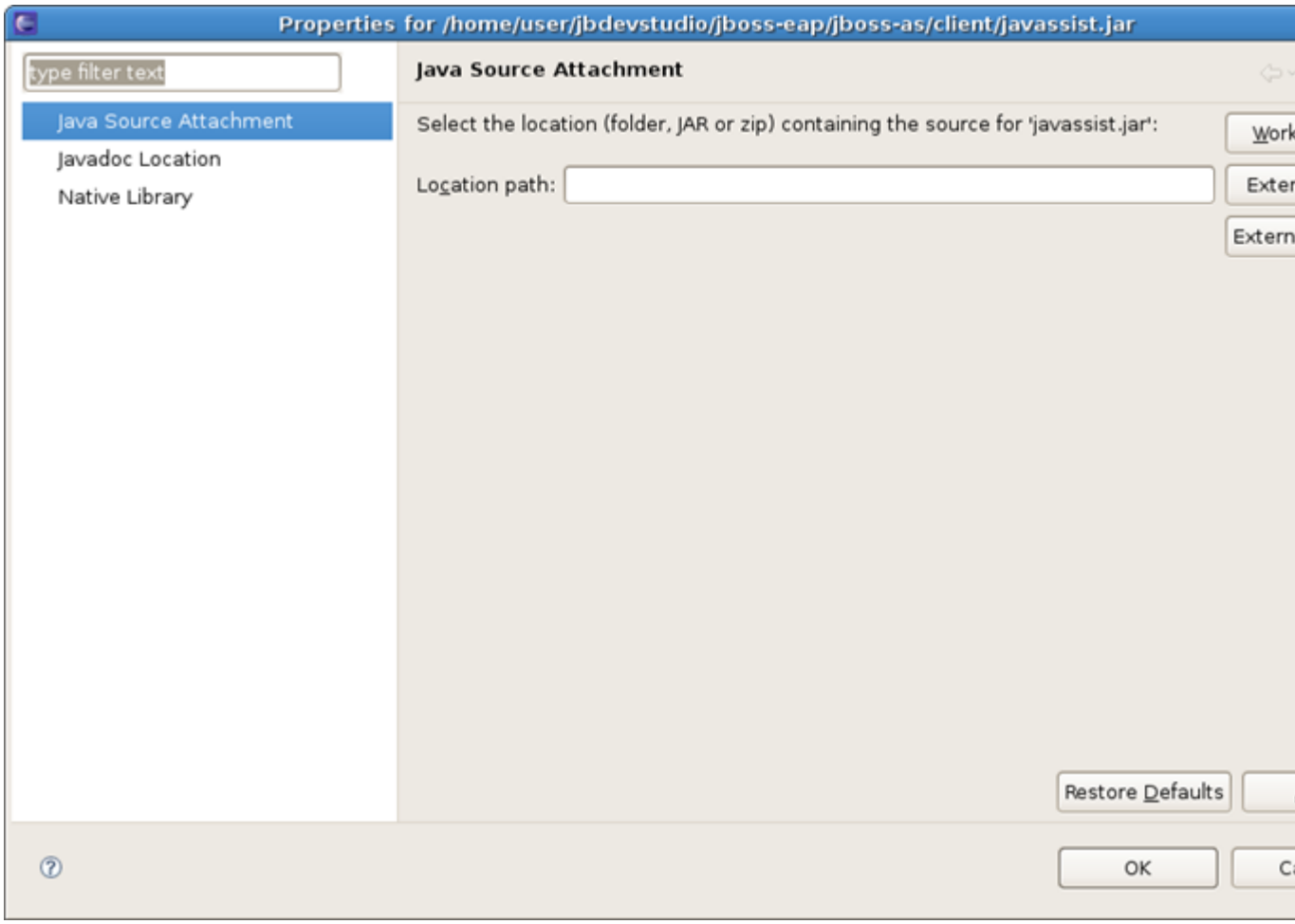

Figure 4.3. Classpath Container: Java Source Attachment

Click on *Apply* and then on *Ok*.

To change JavaDoc Location choose *Javadoc Location* and specify URL to the documentation generated by Javadoc. The Javadoc location will contain a file called *package-list*:

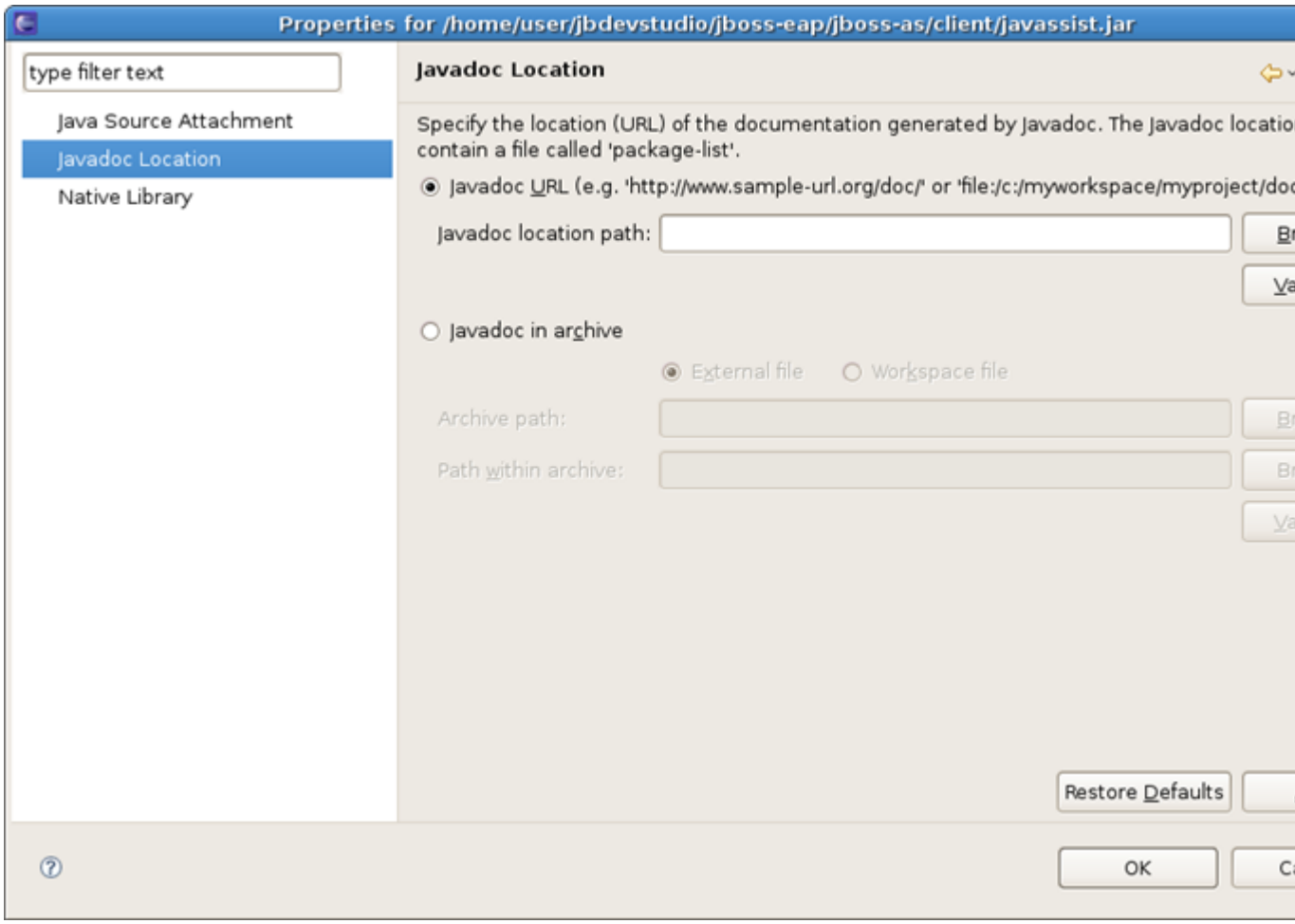

Figure 4.4. Classpath Container: Javadoc Location

Click on *Apply* and then on *Ok*.

4.2. Default Server and Runtime

Open *Window > Preferences > Web Services > Server and Runtime*. On this page, you can specify a default server and runtime.

For ease of use, the better way is to set runtime to JBoss WS.

After server and runtime are specified, click on the Apply button to save the values.

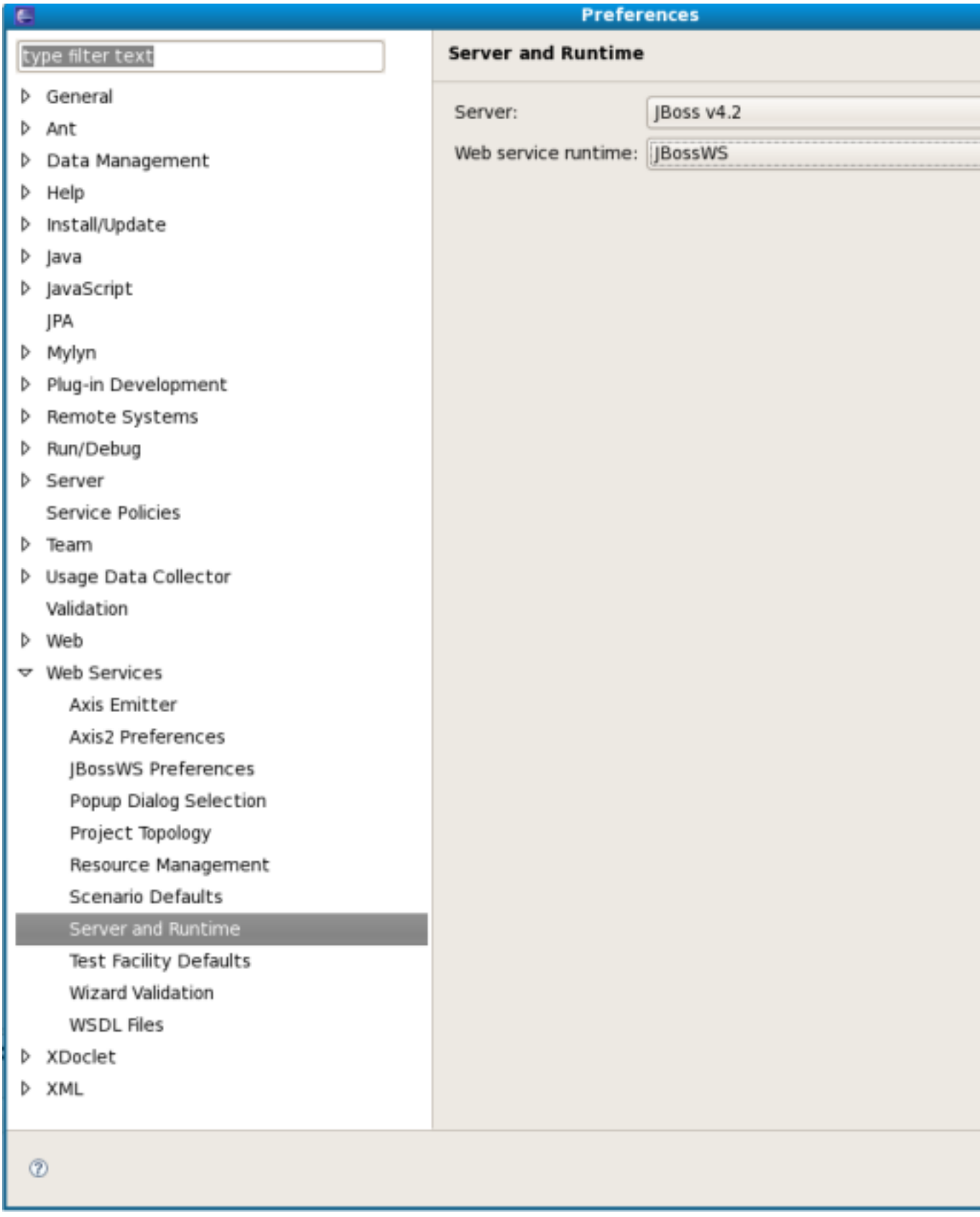
Preferences
type filter text
General
Ant
Data Management
Help
Install/Update
Java
JavaScript
JPA
Mylyn
Plug-in Development
Remote Systems
Run/Debug
Server
Service Policies
Team
Usage Data Collector
Validation
Web
Web Services
Axis Emitter
Axis2 Preferences
JBossWS Preferences
Popup Dialog Selection
Project Topology
Resource Management
Scenario Defaults
Server and Runtime
Test Facility Defaults
Wizard Validation
WSDL Files
XDoclet
XML
Server and Runtime
Server:
JBoss v4.2
Web service runtime:
JBossWS

Figure 4.5.

On the whole, this guide covers the fundamental concepts of work with tooling for JBossWS. It describes how to easily create a Web Service and a Web Service Client using JBossWS Runtime and adjust JBossWS and development environment as well.

If the information on JBossWS tools in this guide isn't enough for you, ask questions on our *forum* [http://www.jboss.com/index.html?module=bb&op=viewforum&f=201]. Your comments and suggestions are also welcome.